# Polyadic Regression and its Application to Chemogenomics

Ioakeim Perros*† Fei Wang‡ Ping Zhang§ Peter Walker¶ Richard Vuduc*
Jyotishman Pathak‡ Jimeng Sun*

**Abstract**

We study the problem of Polyadic Prediction, where the input consists of an ordered tuple of objects, and the goal is to predict a measurement associated with them. Many tasks can be naturally framed as Polyadic Prediction problems. In drug discovery, for instance, it is important to estimate the treatment effect of a drug on various tissue-specific diseases, as it is expressed over the available genes. Thus, we essentially predict the expression value measurements for several (drug, gene, tissue) triads. To tackle Polyadic Prediction problems, we propose a general framework, called Polyadic Regression, predicting measurements associated with multiple objects. Our framework is inductive, in the sense of enabling predictions for new objects, unseen during training. Our model is expressive, exploring high-order, polyadic interactions in an efficient manner. An alternating Proximal Gradient Descent procedure is proposed to fit our model. We perform an extensive evaluation using real-world chemogenomics data, where we illustrate the superior performance of Polyadic Regression over the prior art. Our method achieves an increase of 0.06 and 0.1 in Spearman correlation between the predicted and the actual measurement vectors, for predicting missing polyadic data and predicting polyadic data for new drugs, respectively.
*Keywords:* polyadic prediction, tensors, chemogenomics

## 1 Introduction

Dyadic data are measurements on dyads, i.e., ordered pairs, where each element of a pair originates from a finite set (i.e., data domain) of objects [13]. Typically, those data are represented in a matrix, where a measurement for a pair $(i, j)$ represents some form of relationship between the objects $i$ and $j$ (e.g., user $i$ and movie $j$ are two objects of user and movie data domains, and the corresponding rating reflects a certain relationship between the two objects). We generalize the notion of dyadic data to describe measurements associated with multiple objects (not only pairs), and call the corresponding data *polyadic*.

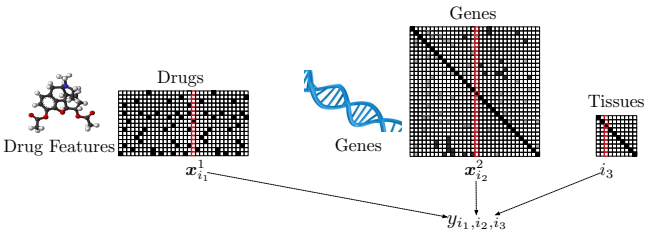



Figure 1: Example use case of Polyadic Prediction. We predict a measurement (expression value) associated with drug $i_1$, gene $i_2$ and tissue $i_3$, indicating whether the drug $i_1$ is effective in treating tissue $i_3$, w.r.t. gene $i_2$. Each drug is described by features capturing its chemical structure, and each gene is described by features reflecting its similarity with other genes.

In this paper, we study the problem of *Polyadic Prediction*, that is, predicting polyadic data. This problem covers many important use cases. For instance, in drug discovery, we are interested in the treatment efficacy of drugs on various types of tissue-specific diseases (e.g., different types of cancer), as this is measured by the expression regarding all the available genes. While thousands of gene expression profiles (i.e., expression values for accessible genes) are available, substantial gaps remain in the combinatorial space across drugs and tissues. The reason for that is mainly the cost associated with producing expression values. An accurate prediction of unknown drug-gene-tissue measurements can empower a better understanding of drug mechanisms, a more precise identification of drug targets (i.e., specific proteins), as well as finding new uses for existing drugs (known as drug repositioning) [23].

We can naturally frame the challenge above as a Polyadic Prediction problem: our data domains correspond to the sets of objects: drugs, genes, and tissues. Given the observed measurements for triads involving a specific object from each data domain, we are interested in predicting unseen triadic data, i.e., unseen measurements for (drug $x$, gene $y$, tissue $z$) triads.

Another notable challenge is to provide accurate predictions for *entirely new objects* from some data domains, which are unseen during the training phase (cold-start problem [29]). For example, instead of directly conducting many expensive lab experiments

*Georgia Institute of Technology
†The work was partially conducted during an internship at Weill Cornell Medicine.
‡Weill Cornell Medicine
§IBM T.J. Watson Research Center
¶Naval Medical Research Center

measuring the expression for a new drug, we could try to estimate its treatment effect first computationally. Then, we could focus on a small subset of targeted drug trials and cut down the corresponding costs.

To enable predictions for new objects, we need some form of external knowledge for their data domain. We illustrate this use case in Figure 1, where we predict the measurement $y_{i_1,i_2,i_3}$ involving drug $i_1$, gene $i_2$ and tissue $i_3$. We assume that feature vectors are available for the drug data domain (e.g., $\boldsymbol{x}^1_{i_1}$ is the feature vector for drug $i_1$). Thus, we try to provide a prediction for $y_{i_1,i_2,i_3}$, even if the drug $i_1$ was not part of any observed measurement during training. In Figure 1, this holds accordingly for the genes' data domain.

Finally, in target applications involving polyadic data, a standard assumption is that significant interactions exist between objects across the data domains. For instance, we assume that the treatment effect of each drug is varying for different gene-tissue combinations. Regarding the example in Figure 1, this property is mathematically reflected through the interactions between drug with gene features and tissue objects. Thus, it is imperative to efficiently integrate those inter-domain interactions, which can be high-order for general polyadic data.

Dyadic Prediction has been studied in recent literature (e.g., [17, 15, 22], also mentioned as bilinear prediction or pair-input/pairwise learning). Still, those works limit their endeavors to dyadic data. Instead, we propose a new framework, which we call *Polyadic Regression*, to address the challenges introduced above. Our main contributions are as follows

- We propose a **general** Polyadic Prediction framework, predicting measurements associated with multiple objects. The fitting algorithm adapts to various tasks (continuous/discrete) and constraints (e.g., sparse solutions via $\ell_1$ norm).
- Our framework is **inductive** [1]: it can incorporate external knowledge to enable predictions for new objects which have not been encountered during training, tackling the so-called *cold-start problem*.
- Our model is **expressive** to fit the needs of Polyadic Prediction: it explores all the high-order interactions across different data domains. Its factorized version is designed to reduce its complexity and prevent overfitting.

We apply the proposed framework to problems in chemogenomics (i.e., chemical genomics). This field is facing many challenges that can be naturally framed as Polyadic Prediction problems, such as drug-induced, cell-specific gene expression prediction and drug-protein interaction prediction of various types [30]. Thus, we conduct an extensive evaluation on both synthetic and real, publicly-available chemogenomics data. The synthetic data experiment establishes the recovery of relevant features, in the presence of interactions. The real data case study on drug-induced, cell-specific gene expression prediction showcases the superior accuracy of Polyadic Regression as compared to the prior art in both of our target use cases. In particular, our approach improves the correlation (standard metric in gene expression analysis [1]) between the predicted and the actual measurement vectors by 0.06 and 0.1, in estimating missing polyadic data and predicting polyadic data involving new drugs, respectively.

[1] Our distinction between the inductive and transductive settings follows the one provided in [15, 22].

## 2 Background

*Tensors* are high-order generalizations of matrices. The *order* of a tensor denotes the number of its modes (e.g., matrices are 2-order tensors). A *fiber* is a vector extracted from a tensor by fixing all modes but one. Considering a $d$-order tensor $\mathcal{S} \in \mathbb{R}^I$, where $I := I_1 \times \cdots \times I_d$, the index set of each individual mode $\mu$ is $I_\mu := \{1, \ldots, n_\mu\}, \mu \in \{1, \ldots, d\}$. *Matricization*, also called *reshaping* or *unfolding*, logically reorganizes tensors into other forms without changing the values themselves. The index set without mode-$\mu$ is $I^{(\mu)} := I_1 \times \cdots \times I_{\mu-1} \times I_{\mu+1} \times \cdots \times I_d$. Then, the *$\mu$-mode matricization* is a mapping from a tensor to a matrix: $\mathbf{S}^{(\mu)} : \mathbb{R}^I \rightarrow \mathbb{R}^{I_\mu \times I^{(\mu)}}$. As a result, the mode-$\mu$ fibers of the tensor become columns of a matrix. Given $\mathbf{U}_\mu \in \mathbb{R}^{J_\mu \times I_\mu}$, the *$\mu$-mode multiplication* (or $\mu$-mode product) is defined by $\mathbf{S} \times_\mu \mathbf{U}_\mu$ and its matricized version is $\mathbf{U}_\mu \mathbf{S}^{(\mu)} \in \mathbb{R}^{J_\mu \times I^{(\mu)}}$. Given matrices $\mathbf{U}_v \in \mathbb{R}^{J_v \times I_v}, v = 1, \ldots, d$, we can generalize this operation for all the tensor modes via the *multi-linear multiplication*, denoted as: $\mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_d \mathbf{U}_d \in \mathbb{R}^{J_1 \times \cdots \times J_d}$ [19].

## 3 Problem & Model Formulation

First, we formally introduce the Polyadic Prediction problem and our proposed model in its most general form. Next, we extend this model to cases when prior knowledge for certain data domains is not available. Then, we present a more efficient, factorized version of it which can easily handle high-order input ($\geq$ 3-order).

**3.1 Problem Definition** Suppose our dataset consists of $N$ samples of the following form: $\{(i_1, i_2, \ldots, i_K), y_{i_1,i_2,\ldots,i_K}\}$, where $y_{i_1,i_2,\ldots,i_K}$ is an observed measurement involving the $K$ objects $i_1, i_2, \ldots, i_K$. We further assume that $i_k \in \{1, 2, \ldots, n_k\}, 1 \leq k \leq K$, that is, the object $i_k$ takes values from a finite set of objects with car-

dinality $n_k$. We call this set the $k$-th data domain, since it contains objects indexed in the $k$-th position of the ordered tuple $(i_1, i_2, \ldots, i_K)$. The goal of Polyadic Prediction is to learn a function to predict the value of unseen measurements $y_{i_1,i_2,\ldots,i_K}$, given the values of the observed ones.

First, in Section 3.2, we assume that external knowledge is available for all $K$ data domains. In that case, each object $i_k$ is described by a feature vector $\boldsymbol{x}^k_{i_k} \in \mathbb{R}^{d_k}$, where the superscript refers to the $k$-th data domain and $d_k$ is the size of its feature vectors. In Section 3.3, we relax this assumption.

**3.2 Core Model** Our core regression model is

$$\begin{aligned} & f(\boldsymbol{x}^1_{i_1}, \boldsymbol{x}^2_{i_2}, \cdots, \boldsymbol{x}^K_{i_K}) \\ & = b + \underbrace{\sum_{k=1}^{K} (\boldsymbol{w}^k)^\top \boldsymbol{x}^k_{i_k}}_{\text{linear terms}} + \underbrace{\sum_{uv} (\boldsymbol{x}^u_{i_u})^\top \mathbf{S}^{uv} \boldsymbol{x}^v_{i_v}}_{\text{dyadic interactions}} \\ & \quad + \underbrace{\sum_{uvr} \mathcal{S}^{uvr} \times_1 \boldsymbol{x}^u_{i_u} \times_2 \boldsymbol{x}^v_{i_v} \times_3 \boldsymbol{x}^r_{i_r}}_{\text{triadic interactions}} + \cdots \\ & \quad + \underbrace{\mathcal{S} \times_1 \boldsymbol{x}^1_{i_1} \times_2 \boldsymbol{x}^2_{i_2} \cdots \times_K \boldsymbol{x}^K_{i_K}}_{\text{general polyadic interactions}} \end{aligned} \tag{3.1}$$

where $b$ is a scalar offset, $\boldsymbol{w}^k \in \mathbb{R}^{d_k}$ is the vector capturing the linear feature effects of each $k$-th data domain, $\mathbf{S}^{uv} \in \mathbb{R}^{d_u \times d_v}$ is the matrix capturing dyadic feature interaction effects across the $u, v$ data domains, $\mathcal{S} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_K}$ is a tensor capturing the $K$-way feature interaction effects across all the data domains. Our core model is expressive enough to capture all the dyadic, triadic, and in general polyadic interactions emerging across features of various data domains.

We assume the matrices/tensors capturing interaction effects are order-independent w.r.t. the data domains, i.e., $\mathbf{S}^{uv}_{ij} = \mathbf{S}^{vu}_{ji}$, $\mathcal{S}^{uvr}_{ijk} = \mathcal{S}^{urv}_{ikj} = \mathcal{S}^{vur}_{jik} = \mathcal{S}^{vru}_{jki} = \mathcal{S}^{ruv}_{kij} = \mathcal{S}^{rvu}_{kji}$ and so on for the higher-order terms. That is, we model the interactions of each group of data domains with a single matrix/tensor, ignoring the rest possible data domain permutations.

**3.3 Partial Induction** The model in Equation (3.1) implies that we have external knowledge available for all the data domains. We call this setting *complete induction*. However, this setting may not hold in practice. For example, we may have drug and gene features available, but no external information describing each one of the tissues. This setting is henceforth referred as *partial induction*.

First, consider a dyadic data example. In complete induction, the measurement for a dyad $(i_1, i_2)$ is given by

$$f(\boldsymbol{x}^1_{i_1}, \boldsymbol{x}^2_{i_2}) = b + \left(\boldsymbol{w}^1\right)^\top \boldsymbol{x}^1_{i_1} + \left(\boldsymbol{w}^2\right)^\top \boldsymbol{x}^2_{i_2} + \left(\boldsymbol{x}^1_{i_1}\right)^\top \mathbf{S}^{12} \boldsymbol{x}^2_{i_2}$$

We now assume that external knowledge is only available for the 1st data domain. In this case, we can predict a measurement associated with any object belonging to the 1st data domain (either it is included in the training set or not) and any (encountered during training) object belonging to the 2nd data domain. One typical analogue for this case is multi-label learning [36], where we have features describing the objects of the 1st, and a set of labels as the objects of the 2nd data domain. We follow the idea proposed in [9] and assume there are shared and individual components in each label predictor. More concretely, the possibility of assigning the label $i_2$ to object $i_1$ can be defined as

$$f_{i_2}(\boldsymbol{x}^1_{i_1}) \quad = \quad b + (\boldsymbol{w}^1)^\top \boldsymbol{x}^1_{i_1} + (\boldsymbol{x}^1_{i_1})^\top \boldsymbol{s}^{12}_{i_2}$$

where $\boldsymbol{w}^1$ is shared across all label predictors, and $\boldsymbol{s}^{12}_{i_2}$ is distinct for each label predictor. If we define $\mathbf{S}^{12} = [\boldsymbol{s}^{12}_1, \boldsymbol{s}^{12}_2, \cdots, \boldsymbol{s}^{12}_{n_2}] \in \mathbb{R}^{d_1 \times n_2}$, then we can re-write $f_{i_2}(\boldsymbol{x}^1_{i_1})$ as

$$f_{i_2}(\boldsymbol{x}^1_{i_1}) \quad = \quad b + \left(\boldsymbol{w}^1\right)^\top \boldsymbol{x}^1_{i_1} + \left(\boldsymbol{x}^1_{i_1}\right)^\top \mathbf{S}^{12} \boldsymbol{e}^2_{i_2}$$

where $\boldsymbol{e}^2_{i_2} \in \mathbb{R}^{n_2}$ is a vector with only the $i_2$-the element being 1, all other elements are zeros (one-hot encoding) and $n_2$ is the cardinality of the 2nd data domain (i.e., number of distinct labels).

We generalize the idea above for the Polyadic Regression model as follows. Let $\mathcal{K} = \{1, 2, \cdots, K\} = \mathcal{K}_U \bigcup \mathcal{K}_F$ be the set of data domains. $\mathcal{K}_U$ is the set of data domains without external knowledge, $\mathcal{K}_F$ is the set of data domains with features describing their corresponding objects. In order to adapt Equation 3.1 to the partial induction setting, we define $\bar{\boldsymbol{x}}^k_{i_k}$ as either the feature vector of object $i_k$ from the $k$-th data domain, when $k \in \mathcal{K}_F$, or the one-hot encoding of size $n_k$ for the $i_k$ object, when $k \in \mathcal{K}_U$, i.e.,

$$\bar{\boldsymbol{x}}^k_{i_k} = \begin{cases} \boldsymbol{x}^k_{i_k} \in \mathbb{R}^{d_k}, & \text{if } k \in \mathcal{K}_F \\ \boldsymbol{e}^k_{i_k} \in \mathbb{R}^{n_k}, & \text{if } k \in \mathcal{K}_U \end{cases}$$

We also use $\bar{\boldsymbol{w}}^k$ to either denote the linear feature effects of the $k$-th data domain when $k \in \mathcal{K}_F$, or a zero vector which eliminates those effects when features are not available (when $k \in \mathcal{K}_U$), i.e.,

$$\bar{\boldsymbol{w}}^k = \begin{cases} \boldsymbol{w}^k \in \mathbb{R}^{d_k}, & \text{if } k \in \mathcal{K}_F \\ \mathbf{0}^{n_k} \in \mathbb{R}^{n_k}, & \text{if } k \in \mathcal{K}_U \end{cases}$$

Then, the model takes the form

$$\begin{aligned} & f(\bar{\boldsymbol{x}}^1_{i_1}, \bar{\boldsymbol{x}}^2_{i_2}, \cdots, \bar{\boldsymbol{x}}^K_{i_K}) \\ & = b + \sum_{k=1}^{K} (\bar{\boldsymbol{w}}^k)^\top \bar{\boldsymbol{x}}^k_{i_k} + \sum_{uv} (\bar{\boldsymbol{x}}^u_{i_u})^\top \bar{\mathbf{S}}^{uv} \bar{\boldsymbol{x}}^v_{i_v} \\ & \quad + \sum_{uvr} \bar{\mathcal{S}}^{uvr} \times_1 \bar{\boldsymbol{x}}^u_{i_u} \times_2 \bar{\boldsymbol{x}}^v_{i_v} \times_3 \bar{\boldsymbol{x}}^r_{i_r} + \cdots \\ & \quad + \bar{\mathcal{S}} \times_1 \bar{\boldsymbol{x}}^1_{i_1} \times_2 \bar{\boldsymbol{x}}^2_{i_2} \cdots \times_K \bar{\boldsymbol{x}}^K_{i_K} \end{aligned} \tag{3.2}$$

If we define $\bar{d}_k$ as

$$\bar{d}_k = \begin{cases} d_k, & \text{if } k \in \mathcal{K}_F \\ n_k, & \text{if } k \in \mathcal{K}_U \end{cases}$$

then $\bar{\mathbf{S}}^{uv} \in \mathbb{R}^{\bar{d}_u \times \bar{d}_v}$, $\bar{\mathcal{S}}^{uvr} \in \mathbb{R}^{\bar{d}_u \times \bar{d}_v \times \bar{d}_r}$, $\bar{\mathcal{S}} \in \mathbb{R}^{\prod_k \bar{d}_k}$.

**3.4 Factorizing Polyadic Interactions** Below, we first assume a complete induction setting, and then extend the discussion for the partial induction one. The core model presented in Equation 3.1 enjoys high expressive power in capturing all high-order interactions across different data domains. However, this approach suffers from high model complexity ($\mathcal{O}(d^K)$ for $K$ data domains and $d$ features per domain), which apart from efficiency issues, may result in a poor generalization performance.

As the pioneering work on Factorization Machines [27] suggested, it is reasonable to assume that feature interactions are not independent, and patterns are emerging among them. Those patterns imply a low-rank structure of the corresponding matrix/tensor reflecting interactions. To incorporate such a structure, one could augment the objective with a rank-minimizing constraint (e.g., trace norm minimization [16]). Still, this approach faces a huge model size problem for high-order interactions and is not feasible for our model.

Instead, we explicitly replace the parameters capturing interaction effects with their low-rank counterparts, i.e., products of sets of low-rank matrices or tensors. Moreover, we take the idea of shared patterns among features [27] a step further: we consider that shared structure also exists among interactions of different orders, so that, for instance, the low-rank approximation of $\mathbf{S}^{uv}$ has common terms with the one of $\mathcal{S}^{uvr}$. We mathematically translate this assumption to having a basis matrix $\mathbf{F}^k \in \mathbb{R}^{d_k \times m}$ capturing the low-rank structure of features for each $k$-th data domain. In that case, we would approximate $\mathbf{S}^{uv}$ as

$$\mathbf{S}^{uv} \approx \mathbf{F}^u (\mathbf{F}^v)^\top = \sum_{j=1}^{m} \mathbf{F}^u_j (\mathbf{F}^v_j)^\top \tag{3.3}$$

where $\mathbf{F}^u_j$ is the $j$-th column of $\mathbf{F}^u$. The formulation above implies that each dyadic effect (entries of $\mathbf{S}^{uv}$) is a result from aggregating the interactions (outer products) among the $m$ feature groups defined by the columns of $\mathbf{F}^u, \mathbf{F}^v$. It is equivalent to the bilinear model proposed in [22].

However, a pure generalization of Equation (3.3) for our model is restrictive. It would imply that the interactions between feature groups share the same contribution (i.e., weighting factor) for various data domain pairs or different orders. For a triadic data example, the weight of the interaction between $\mathbf{F}^u_1$ and $\mathbf{F}^v_1$ towards $\mathbf{S}^{uv}$ is restrained to be the same as the one between $\mathbf{F}^u_1$ and $\mathbf{F}^r_1$ towards $\mathbf{S}^{ur}$. Another restrictive assumption is that the subspaces defined by all the basis matrices $\mathbf{F}^k$ share the same dimension ($m$).

To tackle the above issues, we incorporate a scalar weight for each feature group interaction, capturing its significance towards the specific pair, triplet or higher combination of feature groups considered. We also permit a different dimension for the subspace corresponding to each basis matrix. Thus, Relation 3.3 is transformed to

$$\mathbf{S}^{uv} \approx \mathbf{F}^u \mathbf{C}^{uv} (\mathbf{F}^v)^\top = \sum_{j=1}^{m_u} \sum_{k=1}^{m_v} \mathbf{C}^{uv}_{jk} \mathbf{F}^u_j (\mathbf{F}^v_k)^\top \tag{3.4}$$

so that each dyadic interaction matrix $\mathbf{S}^{uv}$ is approximated by a tri-factorization [5]. For higher-order terms, we adopt the multi-way analogue of tri-factorization, which is the Tucker decomposition [31]:

$$\mathcal{S}^{uvr} \approx \mathcal{C}^{uvr} \times_u \mathbf{F}^u \times_v \mathbf{F}^v \times_r \mathbf{F}^r$$

$$\vdots$$

$$\mathcal{S} \approx \mathcal{C} \times_1 \mathbf{F}^1 \times_2 \mathbf{F}^2 \cdots \times_K \mathbf{F}^K$$

where $\mathcal{C}^{uvr} \in \mathbb{R}^{m_u \times m_v \times m_r}, \mathcal{C} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$ are low-rank core tensors, and $\mathbf{F}^k \in \mathbb{R}^{d_k \times m_k}$ with $m_k \ll d_k$.

The discussion above implies a complete induction setting. Consider now that we have no external knowledge for the $u$-th data domain. In that case, we no longer need to learn a low-rank representation of features $\mathbf{F}^u$. However, it is still useful to learn how the objects from the $u$-th data domain interact with features/objects from other domains. To model this behavior, we fix $\mathbf{F}^u$ to be an identity matrix $\mathbf{I}^u \in \mathbb{R}^{n_u \times n_u}$ and Relation (3.4) becomes

$$\bar{\mathbf{S}}^{uv} \approx \bar{\mathbf{C}}^{uv} (\mathbf{F}^v)^\top$$

where $\bar{\mathbf{S}}^{uv} \in \mathbb{R}^{n_u \times d_v}, \bar{\mathbf{C}}^{uv} \in \mathbb{R}^{n_u \times m_v}$. This is trivially extended for higher-order terms.

## 4 Algorithm

**4.1 Objective Formulation** For simplicity, we assume a complete induction setting and for notational convenience, we use $\{X_i, y_i\}_{i=1}^N$ to represent the $i$-th data sample, where $X_i = (\boldsymbol{x}^1_{i_1}, \boldsymbol{x}^2_{i_2}, \cdots, \boldsymbol{x}^K_{i_K})$ and $y_i$ is the ground truth measurement of the objects involved in $X_i$. We define our objective function as the sum of a term minimizing the desired loss and another one intended to regularize the parameters. This objective can be summarized as

$$\mathcal{L} = \frac{1}{N} \sum_i \underbrace{\ell(f(X_i), y_i)}_{\mathcal{J}(\mathbf{u})} + \lambda \underbrace{\Omega(\{\boldsymbol{w}^k\}_{k=1}^K, \{\mathbf{S}^{uv}\}_{uv}, \cdots, \mathcal{S})}_{\mathcal{R}(\mathbf{u})} \tag{4.5}$$

where $\ell(f(X_i), y_i)$ is the regression loss of function $f$, $\Omega$ is an aggregation of the regularizations imposed on the model parameters and $N$ is the number of training samples.

**4.2 Objective minimization** The most straightforward choice of a method minimizing the objective in Equation (4.5) would be that of an alternating Gradient Descent (GD). Such a choice would suffice in cases when the regularization function is smooth, such as the squared $\ell_2$-norm.

However, this choice falls short for *nonsmooth* regularizers [3], such as the $\ell_1$-norm which is the standard method to induce sparsity in the solution. In those cases, we have to handle the regularizer as a distinct entity, which can possibly be non-differentiable. To do so, we can adopt proximal gradient methods [25] in each alternating iteration, which aim to solve optimization problems of the following form

$$\min_{\boldsymbol{u}\in\mathcal{H}} \mathcal{J}(\boldsymbol{u}) + \mathcal{R}(\boldsymbol{u}) \tag{4.6}$$

where $\mathcal{J}$ is convex and differentiable with Lipschitz continuous gradient, $\mathcal{R}$ is a convex, lower semi-continuous function which is possibly non-differentiable, and $\mathcal{H}$ is some set, typically a Hilbert space. The correspondence between the objectives (4.5) and (4.6) is clear: the loss function corresponds to $\mathcal{J}(\boldsymbol{u})$ and $\mathcal{R}(\boldsymbol{u})$ is considered as the function reflecting any regularizations.

In proximal methods, $\boldsymbol{u}$ minimizes $\mathcal{J}(\boldsymbol{u}) + \mathcal{R}(\boldsymbol{u})$ if and only if $0 \in \partial_{\boldsymbol{u}}(\mathcal{J}(\boldsymbol{u}) + \mathcal{R}(\boldsymbol{u}))$, where $\partial$ is the subdifferential operator. Given a convex function $\psi : \mathcal{H} \to \mathbb{R}$, we can define its proximal operator $\text{prox}_\psi : \mathcal{H} \to \mathcal{H}$ as $\text{prox}_\psi(\boldsymbol{z}) = \arg\min_{\boldsymbol{u}\in\mathcal{H}} \psi(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{z}\|_2^2$, which can be seen as a generalization of a projection [21]. Then, the Proximal Gradient (PG) method [25] dictates the following update rule to solve the objective (4.6)

$$u^{k+1} := \text{prox}_{\gamma\mathcal{R}}\left(u^k - \gamma^k \nabla\mathcal{J}(u^k)\right) \tag{4.7}$$

where $k$ denotes the current iteration and $\gamma^k > 0$ is a step size, which can be found through line search. In the Supplementary Material [2], we include details regarding both the implementation and the various possible choices of losses and regularization functions.

## 5 Experimental Analysis

Due to space limitations, we include our synthetic data experiments in the Supplementary Material. In the following, we address a real-world challenge arising in the field of chemical genomics: the prediction of drug-induced and cell-specific gene expression values.

**5.1 Background** First, we briefly present some background related to our target application. Differential gene expression profiling of *in vitro* drug perturbations refers to the process of measuring the *difference* in gene expression of a certain cell culture (e.g., cells from brain affected with cancer) before and after treating it with a specific drug. A large differential expression value is indicative of a significant change in the cells' behavior, meaning that the drug could potentially treat the corresponding disease. The methodology described above (known as chemogenomic profiling) has provided powerful insights towards several important tasks, such as better understanding of drug mechanisms [32] and drug repurposing [8]. At the same time, we witness an expansion of the publicly available chemogenomic data through the Library of Integrated Cellular Signatures (LINCS) program [6]: they provide drug-induced and cell-specific gene expression measurements for roughly 1000 genes, which are known to be maximally predictive of the expression of the rest of the available genes [6].

As we discussed in Section 1, the data mentioned above have inherently many missing values, since drugs are often measured only for a subset of tissues. Thus, a significant challenge is to estimate the missing expression values. Another important goal is to enable predictions for new drugs, which are unseen during training. The experiments illustrating the superiority of Polyadic Regression in terms of predicting missing values and expression values for new drugs are provided in Sections 5.4 and 5.5, respectively.

**5.2 Formulation for drug effect prediction** Next, we present our formulation towards drug-induced, cell-specific gene expression prediction. Consider that we have $n_1$ drug objects, $n_2$ gene objects and $n_3$ tissue objects. We used external knowledge for the drug and gene data domains, but no knowledge is available for the tissues (partial induction on the domains of drugs and genes). Thus, $\boldsymbol{x}^1_{i_1} \in \mathbb{R}^{d_1}, \boldsymbol{x}^2_{i_2} \in \mathbb{R}^{d_2}$ are the feature vectors for drug $i_1$ and gene $i_2$ respectively. We also incorporate the low-rank assumptions presented in Section 3.4. Thus, the expression value on gene $i_2$ perturbed by drug $i_1$ on tissue $i_3$ is calculated as

$$\begin{aligned} f_{i_3}(\boldsymbol{x}^1_{i_1}, \boldsymbol{x}^2_{i_2}) &= b + (\boldsymbol{w}^1)^\top \boldsymbol{x}^1_{i_1} + (\boldsymbol{w}^2)^\top \boldsymbol{x}^2_{i_2} \\ &+ (\boldsymbol{x}^1_{i_1})^\top \mathbf{F}^1 \mathbf{C}^{12} (\mathbf{F}^2)^\top \boldsymbol{x}^2_{i_2} \\ &+ (\boldsymbol{x}^1_{i_1})^\top \mathbf{F}^1 \bar{\mathbf{C}}^{13} \boldsymbol{e}^3_{i_3} + (\boldsymbol{x}^2_{i_2})^\top \mathbf{F}^2 \bar{\mathbf{C}}^{23} \boldsymbol{e}^3_{i_3} \\ &+ (\bar{\mathcal{C}} \times_1 \mathbf{F}^1 \times_2 \mathbf{F}^2 \times_3 \mathbf{I}^3) \times_1 \boldsymbol{x}^1_{i_1} \times_2 \boldsymbol{x}^2_{i_2} \times_3 \boldsymbol{e}^3_{i_3} \end{aligned} \tag{5.8}$$

where $\mathbf{F}^1 \in \mathbb{R}^{d_1 \times m_1}$, $\mathbf{F}^2 \in \mathbb{R}^{d_2 \times m_2}$, $\mathbf{C}^{12} \in \mathbb{R}^{m_1 \times m_2}$, $\bar{\mathbf{C}}^{13} \in \mathbb{R}^{m_1 \times n_3}$, $\bar{\mathbf{C}}^{23} \in \mathbb{R}^{m_2 \times n_3}$, $\bar{\mathcal{C}} \in \mathbb{R}^{m_1 \times m_2 \times n_3}$, $\mathbf{I}^3 \in \mathbb{R}^{n_3 \times n_3}$ is an identity matrix and $\boldsymbol{e}^3_{i_3} \in \mathbb{R}^{n_3}$ is the one-hot encoding of tissue $i_3$. Since our measurements are continuous, we select the squared loss and solve for the model parameters by optimizing the following objective

$$\begin{aligned} \min_{\substack{\boldsymbol{w}^1, \boldsymbol{w}^2, \mathbf{F}^1, \mathbf{F}^2, \\ \mathbf{C}^{12}, \bar{\mathbf{C}}^{13}, \bar{\mathbf{C}}^{23}, \bar{\mathcal{C}}}} & \frac{1}{N} \sum_{(i_1,i_2,i_3)\in\mathcal{D}} \left(f_{i_3}(\boldsymbol{x}^1_{i_1}, \boldsymbol{x}^2_{i_2}) - y_{i_1,i_2,i_3}\right)^2 \\ &+ \lambda_1\left(\Omega(\boldsymbol{w}^1) + \Omega(\boldsymbol{w}^2)\right) + \lambda_2\left(\Omega(\mathbf{F}^1) + \Omega(\mathbf{F}^2)\right) \end{aligned} \tag{5.9}$$

[2] www.cc.gatech.edu/~iperros3/pdf/sdm17-supp.pdf

where $\mathcal{D}$ stands for our dataset, $N$ for the number of training samples, $\lambda_1$ is the regularization parameter corresponding to the linear feature effects and $\lambda_2$ is the regularization parameter targeting polyadic interactions.

**5.3 Experiment Setup** The version of LINCS data we used comprise 22,412 drugs applied to 56 different tissues for 978 landmark genes. We followed a standard protocol for data pre-processing, which is described in the Supplementary Material. We incorporate external features for the drug and gene data domains. For drugs, we use the substructure fingerprints denoting the chemical structure for each drug [26]. For genes, we use gene-gene similarity features computed using the GOSemSim [35] package in R. In all our experiments, we used the 10 tissues containing the most expression profiles. We also discarded 128 genes for which we did not have any similarity information.

We employ the holdout method to tune the hyper-parameters, so the available samples are split to train, validation and test sets with an approximate ratio of 0.6 : 0.2 : 0.2. The validation set was purely used to tune the hyper-parameters, which are found using logarithmic grid search for each one of the methods. The hyper-parameters achieving the best performance on the validation set were selected and the corresponding performance on the test set is reported. We used $\ell_2$-norm regularization for all the methods.

We used the Spearman's $\rho$ (Spearman rank correlation coefficient ranging from $-1$ to 1) between the vector of predicted and that of true measurements, as the measure of accuracy. This is a standard evaluation metric for gene expression analysis, where correlation metrics are usually preferred over error measures [1].

Polyadic Regression was set to compete with the following approaches:

**Ridge Regression.** A linear regression with L2 regularization without considering interaction terms. We used the *GLMNet* package [10] (Matlab version) implementing Ridge Regression.

**Factorization Machines (FMs)** [27]. FMs are efficiently exploring all the pairwise feature interactions. We used the *libFM* package [28] (C++) implementing FMs and the Monte Carlo Markov Chain (MCMC) fitting algorithm, which is recommended by the author as the least prone to hyper-parameter selection. Thus, other than the rank parameter governing the model size, we only had to tune the standard deviation of the initial distribution of parameters. Note that the regularization-related parameters are automatically determined in MCMC.

**Multi-view machines (MVMs)** [4]. This is another prior work which takes into account both linear and inter-domain interaction terms, and factorizes all of them jointly. We used the *zen* package implementing Multi-view Machines [37, 4] on top of Apache Spark (in local mode) using Scala. The fitting algorithm in this case is Gradient Descent with adaptive gradient (AdaGrad) [7], which is also recommended by the authors in [4], so that the algorithm is insensitive to the choice of the learning rate. We verified the performance boost using AdaGrad and the insensitivity to the initial learning rate in our experiments when using it. Thus, apart from the rank parameter defining the model size, the only hyper-parameter we had to tune was the $\lambda$ value corresponding to $\ell_2$-norm regularization. Note that due to the joint factorization of all the parameters in MVMs, there is a single regularization value to cover the needs of both the linear and interaction terms.

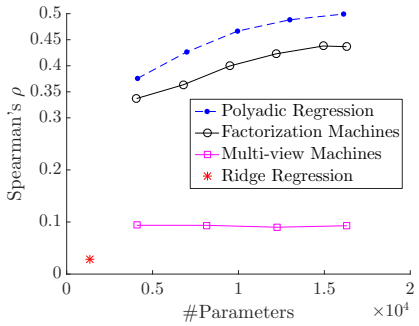



Figure 2: Predicting missing measurements: Spearman correlation between the predicted and the true vector of measurements, as we vary the model size.

To adapt to the supervised learning setting employed by Ridge Regression and FMs, for each sample $\{(i_1, i_2, i_3), y_{i_1,i_2,i_3}\}$, we create a "concatenated" feature vector $[\boldsymbol{x}^1_{i_1}; \boldsymbol{x}^2_{i_2}; \boldsymbol{e}^3_{i_3}] \in \mathbb{R}^{d_1+d_2+n_3}$.

Our method is implemented in Matlab R2015b and C++ with multi-threading capabilities (OpenMP). We used the Mex interface to bridge Matlab and C++ implementation. We also employed the Matlab package *apg* in [24], implementing an accelerated proximal gradient method.

We set all methods other than Ridge Regression to run for a maximum of 2,000 iterations [3]. We do employ early-stopping through cross-validation to avoid overfitting for all methods.

In terms of hardware, we used a server running Ubuntu 14.04 with 251 GB of RAM and 16 physical Intel(R) Xeon(R) E5-2630 CPU's with a maximum clock frequency of 2.40GHz. Each one of the physical cores can exploit 2 threads with hyper-threading enabled.

**5.4 Predicting missing polyadic data** In the following, we evaluate the accuracy of the methods under comparison in predicting missing measurements. Be-

[3] Ridge regression implementation does not require an iteration parameter.

sides picking the top-10 tissues containing the most data, we sub-selected the drugs with measurements available in all of them. Thus, we have 81 drugs for this setting. We can consider that the objects under consideration form a dense tensor containing $81 \times 850 \times 10$ elements denoting the combinations of drugs $\times$ genes $\times$ tissues. Those $688,500$ expression values are split into train, validation and test sets, as explained in Section 5.3. Moreover, we cleaned the initial 881 drug structure features to remove the ones without any variation among the drugs selected. Thus, for this setting, we have 497 drug features kept. The number of gene features is the same as the number of genes since we construct a gene-gene similarity matrix.

We vary the number of parameters for each method (apart from Ridge Regression where the number of parameters is fixed to be the number of features), by tuning the corresponding rank-related parameters. Note that in Polyadic Regression, we have two parameters governing the number of parameters ($m_1$ and $m_2$, as shown in Equation 5.8). We consider that $m_1 = m_2$ and vary them in the range $\{2, 4, 6, 8, 10\}$. Accordingly, we vary the rank-related parameters in FMs and MVMs to reach comparable model sizes.

We present the results of those experiments in Figure 2. We notice that the output of Ridge Regression achieves almost zero correlation with the true measurements. This indicates that the linear terms do not have high predictive towards the predicted outcomes. This is perfectly reasonable considering for instance a specific drug feature (e.g., existence of a certain chemical bond). This feature may target a specific tissue on a certain group of genes, but intuitively there are no drug features that have treating capabilities for every condition. Moreover, the MVMs do not achieve high accuracy either. This approach comes with an efficient model representation which has shown success in recommender systems applications [4]. However, it assumes that the linear and interaction terms are factorized jointly; thus, a single regularization parameter has to cover the needs of both the linear and interaction terms, even if the former ones are almost irrelevant towards the predicted measurements. We believe this is the reason behind the low accuracy of MVMs for this task. Finally, the FMs are promising for this task and the correlation achieved between the output and the measurements' vector reaches a maximum of 0.44. However, they do not take into account 3-order interactions, limiting the model expressiveness for Polyadic Prediction problems.

Polyadic Regression achieves the best performance for all model sizes, reaching a maximum of 0.5 correlation between the output and the true measurements' vector. Note that maximum performance on the validation set was in most cases achieved by setting $\lambda_1$ (in equation 5.9) to be orders of magnitude larger than $\lambda_2$, where the former shrinks the linear and the latter regularizes the interaction effects. Thus, the flexibility of independent representation (and regularization) between linear and interaction terms is crucial towards the target task.

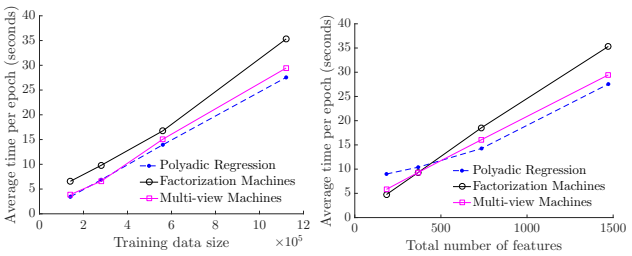
| Method | Spearman's $\rho$ | #Parameters |
|---|---|---|
| Polyadic Regression | $0.23025 \pm 0.0063886$ | 4471 |
| Factorization Machines | $0.1252 \pm 0.0083942$ | 4417 |
| Multi-view Machines | $0.0669 \pm 0.017242$ | 4425 |
| Ridge Regression | 0.0061 | 1473 |

Table 1: Predicting measurements for new drugs: for roughly the same model size, Polyadic Regression achieves 0.1 increase in Spearman correlation between the predicted and the true vector of measurements, against the best-performing competing method. Results are averaged over 5 runs and standard deviation is reported.

**5.5 Predicting polyadic data for new drugs** We also tackle another important challenge in the field of chemogenomics: predicting expression values for new drugs, unseen during the training phase. First, we want to incorporate more drugs than the ones used in Section 5.4, so that the task becomes more challenging. Thus, instead of constraining the drugs to have relevant data in all the 10 tissues, we require that they have data in at least 2 out of them. This requirement is fulfilled by $2,169$ drugs, which we further randomly subsample to 500. The number of relevant drug features (non-zero variation) is 612 in this case. In sum, for the current task, we consider 500 drugs, 850 genes and 10 tissues. In contrast to Section 5.4, we only have $\approx 44\%$ of the measurements among the object combinations available, thus leading to $1,870,850$ data samples. Since we are only interested in predictions for new drugs (predicting for new genes is not a practical use case and we have no external knowledge for tissues), we follow the holdout method by constraining though that the the train, validation and test sets have no common drugs.

We evaluate the robustness of the competing approaches, thus we fixed Polyadic Regression, FMs and MVMs to roughly the same model size and run the experiments 5 times, reporting average performance and standard deviation. We provide the results in Table 1. Polyadic Regression achieves 0.1 increase in correlation between the predicted and the true vector of measurements, against the best-performing competing method. Moreover, we remark its robustness in terms of different initialization of parameters.

**5.6 Scalability** We assess the scalability properties of the methods under comparison w.r.t. varying sizes of training data and input features. To do so, we used the data described in Section 5.5. Regarding the scalability for increasing number of training examples, we constructed smaller sets of data, by including random




(a) Training Data Size (b) Input Feature Dimension

Figure 3: The scalability in terms of training time per epoch with respect to increasing training data size and input feature dimension.

subsets of $1/8, 1/4, 1/2$ of the total samples. As concerns the scalability for increasing number of input features, we fixed the number of training examples and removed an equal number of features from the drug and gene domains, so as to reach the desired total feature number. We measured the training time per epoch, as an average over 100 epochs.

Note that the *GLMNet* package implementing Ridge Regression uses a cyclical coordinate descent algorithm, updating a specific parameter in each iteration. This is in contrast to the rest of the methods which update the whole space of parameters; thus, a comparison in terms of time per epoch with Ridge Regression would not be meaningful and is not included. However, we empirically remark that it is the fastest method, which is reasonable considering it only takes into account linear feature effects.

We provide the results in Figures 3(a) and 3(b). We remark that all methods share similar (near-linear) scalability properties. We would like to emphasize that a direct time comparison between different methods would not be fair, since they are implemented using different setup (single-threaded, multi-threaded).

## 6 Related Work

Polyadic Regression essentially tackles the general case of the Dyadic Prediction problem which is studied in [17, 20, 15, 22]. The corresponding approaches though, only predict dyadic and cannot generalize to predicting polyadic data. Other works that also model only dyadic interactions are the Sparse Factorization Machine [34] and the Conditional High-Order Boltzmann Machine [14].

Multi-view Machines [4] (MVMs) is a recently proposed model, exhibiting success in recommender systems' applications. MVMs capture both linear and higher-order interactions across features of different data domains, and *jointly* factorize all of them by a CP tensor decomposition [12]. On one hand, this decision limits the number of parameters to learn. However, it restricts to a target model where the linear and interaction terms are composed from the same low-rank factors, and have to share the same regularization. This may be a limiting factor when the interaction and linear terms have very different contributions towards the predicted measurements. Our model is more flexible, allowing for a different treatment among the linear and interaction terms.

As concerns other lines of work, the notion of multi-view learning, as it has been hitherto used in the literature [33], does not tackle the challenges we introduced in Section 1. It is limited to models either accepting only two inputs [18] or learning correlations at the view-level (as in multiple-kernel learning [11]) and not between features of different representations, thus limiting the model's expressiveness and interpretation potential. On a different note, while tensor regression methods (e.g., [2]) predict polyadic data by generally exploring only the highest order of possible interactions, they cannot provide predictions for new objects (e.g., new drugs in our target application), in the post-training phase.

## 7 Conclusions

We proposed Polyadic Regression, a general framework predicting measurements associated with multiple objects. Our framework enables predictions for new objects, unseen during training, thus tackling the so-called cold-start problem. Our model is expressive for addressing general Polyadic Prediction problems, by exploring all the high-order interactions across different data domains, in an efficient, factorized, way. We evaluate our approach with real chemogenomics data, demonstrating its superior accuracy over the prior art. As future work, we plan to apply Polyadic Regression to other fields and further improve its scalability.

## Acknowledgments

This work was supported by NSF grants under number IIS-1418511 and CCF-1533768, research partnership between Children's Healthcare of Atlanta and the Georgia Institute of Technology, Google Faculty Award and UCB. Portions of this research were supported by Defense Medical Research Program Grant DM130137. The opinions of the authors do not necessarily reflect those of the United States Navy. PB Walker is a military service member. This work was prepared as part of their official duties. Title 17 U.S.C. 101 defines U.S. Government work as a work prepared by a military service member or employee of the U.S. Government as part of that person's official duties. The work of Fei Wang is partially supported by NSF grant under number IIS-1650723. The first author would like to thank Mohammad Taha Bahadori for his valuable feedback which improved the paper's clarity.

## References


[1] David B Allison et al. *DNA microarrays and related genomics techniques: design, analysis, and interpretation of experiments*. CRC Press, 2005.

[2] Mohammad Taha Bahadori et al. "Fast multivariate spatio-temporal analysis via low rank tensor learning". In: *NIPS*. 2014, pp. 3491–3499.

[3] Léon Bottou et al. *Optimization Methods for Large-Scale Machine Learning*. Tech. rep. arXiv:1606.04838, 2016. URL: http://leon.bottou.org/papers/tr-optml-2016.

[4] Bokai Cao et al. "Multi-view Machines". In: *WSDM 2016*, pp. 427–436.

[5] Chris Ding et al. "Orthogonal nonnegative matrix tri-factorizations for clustering". In: *KDD 2006*, pp. 126–135.

[6] Qiaonan Duan et al. "LINCS Canvas Browser: interactive web app to query, browse and interrogate LINCS L1000 gene expression signatures". In: *Nucleic acids research* 42.W1 (2014), W449–W460.

[7] John Duchi et al. "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[8] Joel T Dudley et al. "Exploiting drug–disease relationships for computational drug repositioning". In: *Briefings in bioinformatics* 12.4 (2011), pp. 303–311.

[9] Theodoros Evgeniou and Massimiliano Pontil. "Regularized multi–task learning". In: *KDD*. ACM. 2004, pp. 109–117.

[10] Jerome Friedman et al. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. URL: http://www.jstatsoft.org/v33/i01/.

[11] Mehmet Gönen and Ethem Alpaydın. "Multiple kernel learning algorithms". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2211–2268.

[12] Richard A Harshman. "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis". In: *UCLA Working Papers in Phonetics* (1970).

[13] Thomas Hofmann et al. "Learning from dyadic data". In: *Advances in neural information processing systems* (1999), pp. 466–472.

[14] Yan Huang et al. "Conditional High-Order Boltzmann Machine: A Supervised Learning Model for Relation Learning". In: *ICCV*. 2015, pp. 4265–4273.

[15] Prateek Jain and Inderjit S Dhillon. "Provable inductive matrix completion". In: *arXiv preprint arXiv:1306.0626* (2013).

[16] Shuiwang Ji and Jieping Ye. "An accelerated gradient method for trace norm minimization". In: *ICML*. ACM. 2009, pp. 457–464.

[17] Bo Jin et al. "Multitask Dyadic Prediction and Its Application in Prediction of Adverse Drug-Drug Interaction". In: *AAAI 2017 (to appear)*.

[18] Sham M Kakade and Dean P Foster. "Multi-view regression via canonical correlation analysis". In: *International Conference on Computational Learning Theory*. Springer. 2007, pp. 82–96.

[19] Tamara G Kolda and Brett W Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[20] Aditya Krishna Menon and Charles Elkan. "A log-linear model with latent features for dyadic prediction". In: *ICDM*. IEEE. 2010, pp. 364–373.

[21] Jean-Jacques Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *CR Acad. Sci. Paris Sér. A Math* 255 (1962), pp. 2897–2899.

[22] Nagarajan Natarajan and Inderjit S Dhillon. "Inductive matrix completion for predicting gene–disease associations". In: *Bioinformatics* 30.12 (2014), pp. i60–i68.

[23] Kerry A O'Connor and Bryan L Roth. "Finding new tricks for old drugs: an efficient route for public-sector drug discovery". In: *Nature reviews Drug discovery* 4.12 (2005), pp. 1005–1014.

[24] Brendan O'Donoghue. *apg*. https://github.com/bodono/apg. 2016.

[25] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends in optimization* 1.3 (2013), pp. 123–231.

[26] *PubChem Substructure Fingerprints*. https://pubchem.ncbi.nlm.nih.gov/. Accessed: 2016-09-19.

[27] Steffen Rendle. "Factorization machines". In: *ICDM*. IEEE. 2010, pp. 995–1000.

[28] Steffen Rendle. "Factorization machines with libfm". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3 (2012), p. 57.

[29] Mohit Sharma et al. "Feature-based factorized bilinear similarity model for cold-start top-n item recommendation". In: *SDM*. Vol. 15. SIAM. 2015, pp. 190–198.

[30] Marina Sirota, Joel T Dudley, et al. "Discovery and preclinical validation of drug indications using compendia of public gene expression data". In: *Science Translational Medicine* 3.96 (2011), 96ra77–96ra77.

[31] Ledyard R Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311.

[32] Guo Wei, David Twomey, et al. "Gene expression-based chemical genomics identifies rapamycin as a modulator of MCL1 and glucocorticoid resistance". In: *Cancer cell* 10.4 (2006), pp. 331–342.

[33] Chang Xu et al. "A survey on multi-view learning". In: *arXiv preprint arXiv:1304.5634* (2013).

[34] Jianpeng Xu et al. "Synergies that Matter: Efficient Interaction Selection via Sparse Factorization Machine". In: *SDM 2016*, pp. 108–116.

[35] Guangchuang Yu, Fei Li, et al. "GOSemSim: an R package for measuring semantic similarity among GO terms and gene products". In: *Bioinformatics* 26.7 (2010), pp. 976–978.

[36] Min-Ling Zhang and Zhi-Hua Zhou. "A review on multi-label learning algorithms". In: *Knowledge and Data Engineering, IEEE Transactions on* 26.8 (2014), pp. 1819–1837.

[37] Bo Zhao et al. *Zen*. https://github.com/cloudml/zen. 2016.